

REMARKS

7/23/03

Claims 10-26 are pending in the present application. No amendments to the claims are made by this Response. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 103, Alleged Obviousness, Claims 10-13 and 16-26

The Office Action rejects claims 10-13 and 16-26 under 35 U.S.C. § 103 as being unpatentable over Gong (U.S. Patent No. 6,192,476) in view of Bak et al. (U.S. Patent No. 6,009,517). This rejection is respectfully traversed.

With regard to independent claim 10, the Office Action states:

As to claim 10, Gong teaches a Privilege ("...permissions..." Col. 11 Ln. 38-67), Method (Check Permission Method 382 Method 340-3 Method 340-2 Method 340-1 Col. 11 Ln. 1-35), Data Processing system (Computer System 100 Col. 5 Ln. 26-67), a Current Thread (Thread 306 Col. 10 Ln. 66-67, Col. 11 Ln. 1-35), a Run-Time Environment (Code Executor 210 Col. 6, Ln. 50-67), a Stack (Call Stack 308 Col. 10 Ln. 66-67, Col. 11 Ln. 1-35), a Stack Frame (Frame 310-F Frame 310-4 Frame 310-3 Frame 310-2 Frame 310-1 Col. 11 Ln. 1-35) a Thread Identifier (Thread 306 Col. 15 Ln. 21-67, Col. 16 Ln. 1-67: NOTE: Thread 306 includes a thread identifier), locating a linked List and searching the linked list (Stack 308 is queue/linked list of frames and the steps (Steps 430, 440, 450, 490) of inspecting/traversing the stack) and a Stack Frame Extension (Col. 11 Ln. 16-18). Gong is silent with reference to a stack frame pointer.

Bak teaches a Stack Frame Pointer ("...frame pointer..." Col. 7 Ln. 1-36). It would have been obvious to apply teaching of Bak to the system of Gong. One would have been motivated to make such a modification in order to traverse the frames on the execution stack.

Office Action dated November 7, 2002, pages 2-4.

Claim 10, which is representative of independent claims 19 and 26 with regard to similarly recited subject matter, reads as follows:

10. A process for providing a privilege for a method of a current thread that is currently executing in a run-time environment in a data processing system, the run-time environment having a stack comprising stack frames with stack frame pointers for associated methods, the process comprising the computer-implemented steps of:

using a thread identifier of the current thread, locating a linked list;
and
searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method, wherein an entry of the linked list is a stack frame extension. (emphasis added)

Applicants respectfully submit that neither Gong nor Bak, either alone or in combination, teach or suggest locating a linked list using a thread identifier of a current thread or to searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of a method, wherein an entry of the linked list is a stack frame extension.

Gong is directed to a system for controlling access to computer resources. The System of Gong is directed to determining whether a thread may access a particular resource. With the system of Gong, a thread may access a particular resource only if at least one permission of a protection domain associated with each method in the call hierarchy of the thread encompasses the permission required to perform the requested action. An exception is provided in the form of a privilege flag that may be associated with a method in the call hierarchy.

Gong does teach that the call hierarchy, or call stack, is comprised of frames and that the Check Permission method traverses the call stack and determines if each method in the call stack enables the privilege (see Figure 4 of Gong). However, Gong does not teach or suggest using a thread identifier to locate a linked list or searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method for which a privilege is being provided. This is because Gong is not interested in locating a particular entry in the linked list that matches an entry for the method for which a privilege is to be provided, but rather merely traverse the call stack each time an access request is received to determine if all of the methods in the call stack enable the privilege.

The Office Action alleges that Gong teaches locating a linked list using a thread identifier merely because Gong allegedly teaches a thread identifier in thread 306. It should first be noted that nowhere in Gong is it ever mentioned that the thread 306 has a thread identifier. The Office Action alleges that the thread identifier is mentioned in column 15, lines 21-67 and column 16, lines 1-67 which read as follows:

The method shown in FIG. 4 shall now be described with reference to the thread 306 and stack 308 illustrated in FIG. 3. Assume for example, that thread 306 is executing a user interface method 340-1 to update a password. To update the password, thread 306 then invokes method 340-2 (the method to update a password), then method 340-3 (the method to update a file). Assume further that method 340-2 is privileged.

In step 410, the request for a determination of whether an action is authorized is received. After detecting that a request for an action to update the password file has been detected by invoking method 340-3, the permission required to perform the action is determined.

In the present example, the action is updating the password file and the required permission to perform the action is "write" to file "d:/sys/pwd". A request is made to determine whether the action is authorized by invoking the check permission method 382 of the access controller 380, passing in as a parameter the permission required to perform the action. Note the current state of call stack 308 is shown in FIG. 3.

Steps 430, 440, 450 and 460 define a loop in which permissions associated with the methods in the call stack are checked. The loop continues until a privileged method is encountered, or all of the methods in the call stack of have been checked. For the purposes of explanation, the method whose privileges are currently being checked is referred to as the "selected method".

In step 430, a determination is made as to whether one of the permissions associated with the selected method encompasses the permission required. The permissions associated with a method are the permissions associated with the protection domain that is associated with the method. If the determination made in step 430 is that a permission associated with the selected method encompasses the permission required, control passes to step 440.

During the first iteration of the loop, the frame that immediately precedes the frame associated with the check permission method of the access controller is inspected. In this example, the frame associated with the check permission method 382 is frame 310-4. The frame that immediately precedes frame 310-4 is frame 310-3. Consequently, during the first iteration of the loop, frame 310-3 will be inspected. Frame 310-3 is associated with method 340-3, which is associated with protection domain 250-3. Because a permission associated with protection domain object 250-3 (permission to "write" to "d:/sys/*") encompasses the permission required (i.e. "write" to "d:/sys/pwd"), control passes to step 440.

In step 440, a determination is made of whether the invocation of a selected method represents the enabling invocation. This determination is based on the privilege flag 312 of frame 310 corresponding to the invocation of the selected method. If the determination is that the invocation of the selected method does not represent the enabling

invocation, control passes to step 450. In this example, the privilege status of frame 310-3 is not set to indicate that the frame represents the enabling invocation, thus control passes to step 450.

In steps 450, the next method is selected. The next method is the method corresponding to frame below the current frame based on the calling hierarchy represented by call stack 308. In this example, the frame below the current frame 310-3 is frame 310-2. The method corresponding to frame 310-2 is method 340-2.

In step 460, a determination is made of whether a method was selected in step 450. If a method was selected, control reverts to step 430.

In the current example, control passes to step 430 because method 340-2 was selected. In step 430, the determination made is that the protection domain associated with method 340-2 (protection domain object 250-2) includes a permission encompassing the permission required ("write" to "d:/sys/pwd") because a permission associated with protection domain object 250-2 ("write" to "d:/sys/pwd") explicitly encompasses the permission required. Control passes to step 440.

In step 440, the determination made is that the invocation of a selected method represents the enabling invocation because the privilege flag 312 indicates that the invocation corresponding to frame 310-2 is an enabling invocation. A message is transmitted indicating the permission request is valid. Then, performance of the steps ends.

Note that by exiting the performance of the steps at step 440 when the selected method represents the enabling invocation, the authorization of the requested action is based on the privileged protection domain and any protection domains associated with methods invoked after the invocation of the enabling invocation.

Assume in the current example that the privilege mechanism was never invoked. Thus in step 440, the determination made is that invocation of a selected method does not represent the enabling invocation because the privilege flag 312 indicates that the invocation corresponding to frame 310-2 is not an enabling invocation.

In steps 450, the next method selected is method 340-1 because the frame below the current frame 310-2 is frame 310-1 and the method corresponding to frame 310-1 is method 340-1. In step 460, the determination made is that a next method was selected in step 450, thus control reverts to step 430.

In step 430, the determination made is that the protection domain associated with method 340-1 (protection domain object 250-1) does not include the permission required ("write" to "d:/sys/pwd") because no permission associated with protection domain object 250-1 (i.e. "c:/tmp" the only permission associated with the protection domain) encompasses the permission required. Control then passes to step 490.

In step 490, a message indicating that the requested action is not authorized is transmitted. In embodiment of the invention, the message is transmitted by throwing an exception error.

The term "thread identifier" never appears in either of these sections and there is no other element described in these sections that would perform the function of a thread identifier. However, even if Gong were interpreted to teach a thread identifier, Gong still does not teach or suggest locating a linked list based on a thread identifier of a current thread. While Gong teaches that a thread may have a call stack, there is no teaching in Gong to actually perform the function of locating a call stack based on a thread identifier. All that is taught in Gong is to traverse the call stack of the thread in order to determine if every method of the call stack enables the privilege that is required to perform a requested action.

The Office Action further alleges that Gong teaches searching the linked list in Figure 4 with specific reference to steps 430, 440, 450 and 490. While Figure 4 illustrates the Check Privilege method traversing the call stack to determine if all of the methods in the call stack enable the required privilege, Figure 4 does not teach or suggest searching the call stack for a particular entry having a stack frame pointer that matches the stack frame pointer of a method for which a privilege is to be provided. The call stack is merely traversed by the operation of Figure 4, it is not searched. This is clear in that the result that is generated by way of the operation of Figure 4 is not an entry in a linked list that has a stack frame pointer that matches the stack frame pointer of a method for which a privilege is to be provided. Rather, the result is an indication as to whether all of the methods in the call stack enable the privilege or not.

Thus, Gong does not teach or suggest either of the steps of claim 10. Moreover, Bak does not provide for the deficiencies of Gong. Bak is directed to a system that enables frames for code written in different languages to be added to the same call stack. As illustrated in Figure 5 of Bak, a prior art call stack includes frames that have a first portion that identifies a return address, a second portion that identifies state variables, local variables and operands, and a third portion that includes a frame pointer to point to the next frame in the call stack. With the system of Bak, as illustrated in Figure 7, the call stack includes Java frames, and a block of data identifying an entry frame of the call stack, the last java stack pointer and the last java frame pointer that point to the next Java frame in the call stack. A frame of a different type may be placed between Java frames

as illustrated. Thus, similar to the frame pointers in Figure 5, the block in Figure 7 allows the call stack to be traversed with regard to Java frames.

As with Gong above, there is no teaching or suggestion in Bak to identify a linked list using a thread identifier. Similar to Gong, Bak merely teaches that a thread may have an associated call stack. There is nothing in Bak that teaches to identify a particular call stack based on a thread identifier.

In addition, Bak does not teach or suggest searching a linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method for which a privilege is to be provided. While Bak teaches stack frame pointers generally and that stack frame pointers may be used to traverse the call stack, there is no teaching in Bak to search for a specific entry in the call stack that has a stack frame pointer that matches the stack frame pointer of a particular method. The general teaching of a stack frame pointer in Bak does not obviate the specific feature of searching a linked list for an entry having a stack frame pointer that matches the stack frame pointer of a particular method. Furthermore, the Office Action fails to show where Bak teaches or suggests this feature because Bak is only relied upon as generally teaching a stack frame pointer.

Moreover, neither Gong nor Bak teach or suggest that the linked list entries are stack frame extensions. To the contrary, the call stacks in both Gong and Bak are call stacks that contain frames, not stack frame extensions. Thus, neither reference teaches or suggests an entry of the linked list being a stack frame extension.

The Office Action alleges that Gong teaches this feature at column 11, lines 16-18, however this section of Gong merely states "When a method is invoked, a frame corresponding to the method is added to the top of the call stack 308." This describes a call stack in which stack frames are added when a method is invoked. This does not describe a linked list in which entries are stack frame extensions.

Thus, in view of the above, Applicants respectfully submit that neither Gong nor Bak, either alone or in combination, teach or suggest the features of independent claim 10. Independent claims 19 and 26 recite similar features to those in claim 10 but in system and computer program product format, respectively. Therefore, claims 19 and 26 are considered to define over the alleged combination of Gong and Bak for similar reasons as noted above.

With regard to independent claim 18, the Office Action fails to address the specific features set forth in claim 18. Rather, the Office Action merely states "As to claim 18, see the rejection of claim 10." Claim 18 contains different features from those in claim 10 – features not addressed by the rejection of claim 10. For example, claim 18 recites "a set of stack frame extensions, wherein a stack frame extension comprises: a pointer to a stack frame for a method; a data field for privilege data for the method; a data field for validation data for the method." None of these fields of a stack frame extension are addressed by the Office Action and there is no showing where either of Gong or Bak allegedly teach these features. This is because there is no teaching or suggestion in Gong or Bak with regard to these fields of a stack frame extension.

In view of the above, Applicants respectfully submit that neither Gong nor Bak, either alone or in combination, teach or suggest the features of independent claims 10, 18, 19 and 26. At least by virtue of their dependency on claims 10 and 19, respectively, neither Gong nor Bak, either alone or in combination, teach or suggest the features of dependent claims 11-13, 16-17 and 20-25. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 10-13 and 16-26 under 35 U.S.C. § 103(a).

Furthermore, neither Gong nor Bak, either alone or in combination, teach or suggest the specific features recited in dependent claims 11-13, 16-17 and 20-25. For example, neither reference teaches or suggests locating the linked list within a stack frame shadow apparatus comprising a plurality of linked lists, each linked list of the plurality of linked lists being associated with a thread, as recited in claim 11 and a similar feature in claim 20. With regard to this feature, the Office Action alleges that "although a plurality of linked list is not explicitly taught, Gong does teach that a parent thread could created child thread that inherits the parent's call stack." While this may be true, inheriting a call stack in no way teaches locating a linked list within a stack frame shadow apparatus comprising a plurality of linked lists, each linked list of the plurality of linked lists being associated with a thread. There simply is no teaching in either reference regarding a shadow apparatus having a plurality of linked lists, each one associated with a thread. Thus, there cannot be any teaching in these references of locating a linked list within such a shadow apparatus.

Regarding claims 12 and 21, these claims recite features similar to those features recited in claim 18 noted above. Therefore, claims 12 and 21 are also distinguished over the alleged combination of Gong and Bak because neither reference teaches or suggests a stack frame extension that comprises a stack frame pointer to a method, privilege information and validation information. The Office Action alleges that claim 10 meets claim 12 except for privilege information and validation information and alleges that Gong teaches privilege information and validation information. As noted above with regard to claim 18, neither Gong nor Bak teach or suggest a stack frame extension, let alone a stack frame extension that comprises a stack frame pointer to a method, privilege information and validation information. While Gong may generally teach the use of privileges and validation, there is no teaching or suggestion in Gong to include privilege information and validation information in a stack frame extension along with a stack frame pointer to a method.

With regard to claims 13 and 22, the Office Action merely references the rejection of claim 12. Again, neither Gong nor Bak teach the features of claims 12 and 21 as discussed above. In addition, the Office Action has failed to address the specific features of claim 13, i.e. validation information comprising a name of a method, a signature of a method, or a return address of a method. Neither reference teaches a stack frame extension that includes validation information comprising these features.

Regarding claims 16 and 25, neither reference teaches or suggests retrieving privilege information and validation information for a matching entry from a linked list if a matching entry is found in response to a request to retrieve privileges for the method. The Office Action alleges that Gong teaches retrieving privilege information and validation information at step 440 and column 16, lines 31-43. Step 440 of Gong is a determination as to whether a method enables a privilege. Column 16, lines 31-43 of Gong teaches that "the invocation of a selected method represents the enabling invocation because the privilege flag indicates that the invocation corresponding to frame 310-2 is an enabling invocation." Neither step 440 nor the text at column 16, lines 31-43 teach or suggest retrieving privilege information and validation information for a matching entry from a linked list if a matching entry is found in response to a request to retrieve privileges for the method. Merely determining if a method enables a privilege does not

teach all of the specific features recited in claims 16 and 25. Furthermore, Bak does not provide any teaching or suggestion regarding these features of claims 16 and 25 either.

Regarding claim 17, neither reference teaches or suggests storing privilege information in a stack frame shadow apparatus, querying a stack frame shadow apparatus for privilege information for a method, or deleting privilege information in a stack frame shadow apparatus to revert a privilege for a method. As previously mentioned above with regard to claims 11 and 20, neither reference teaches a stack frame shadow apparatus. Furthermore, the Office Action fails to show where either reference teaches a stack frame shadow apparatus. Instead, the Office Action alleges that Gong teaches that the steps of Figure 4 could be applied to a parent/child thread and that somehow this is the same as a stack frame shadow apparatus. Applicants respectfully submit that there is no correspondence between anything in Figure 4 of Gong, or the alleged ability of a child thread to inherit the call stack of a parent thread, that would render obvious a stack frame shadow apparatus. Moreover, since neither reference teaches or suggests a stack frame shadow apparatus, neither reference could be found to teach or suggest storing privilege information in a stack frame shadow apparatus, querying a stack frame shadow apparatus, or deleting privilege information from a stack frame shadow apparatus, as recited in claim 17.

With regard to claims 23 and 24, the Office Action states "see the rejection of claim 14" and "see the rejection of claim 15." It should be noted that claims 14 and 15 are not rejected based on a combination of only Gong and Bak. Rather, claims 14 and 15 are rejected based on Gong, Bak and Introduction to the Capabilities Classes (see below). However, to the extent that the Office Action intended to reject claims 23 and 24 based on the alleged combination of Gong and Bak, neither reference, either alone or in combination, teaches or suggests adding means for adding an entry to the linked list if no matching entries are found in response to a request to enable a privilege for the method (claim 23) or removing means for removing a matching entry from the linked list if a matching entry is found in response to a request to revert a privilege for the method (claim 24). The Office Action admits that Gong and Bak do not teach these features in the rejection of claims 14 and 15, discussed hereafter. Furthermore, as set forth hereinbelow with regard to claims 14 and 15, the Introduction to the Capabilities Classes

reference also does not teach or suggest this feature. Claims 23 and 24 distinguish over Gong, Bak and Capabilities Classes for similar reasons as noted hereafter with regard to claims 14 and 15.

II. 35 U.S.C. § 103, Alleged Obviousness, Claims 14 and 15

The Office Action rejects claims 14 and 15 under 35 U.S.C. § 103(a) as being unpatentable over Gong and Bak, and further in view of Introduction to the Capabilities Classes (Hereinafter referred to as ICC pages 1-15). This rejection is respectfully traversed.

Gong and Bak have been discussed above. ICC does not provide for the deficiencies of these references. Specifically, ICC does not teach or suggest using a thread identifier of a current thread to locate a linked list and searching the linked list for an entry having a stack frame pointer that matches the stack frame pointer of the method, as recited in claim 10 from which claims 14 and 15 depend. Therefore, claims 14 and 15 define over the alleged combination of Gong, Bak and ICC. Accordingly, Applicants respectfully request withdrawal of the rejection of claims 14 and 15 under 35 U.S.C. § 103(a).

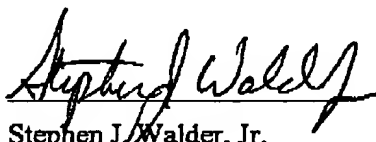
7/23/03

III. Conclusion

It is respectfully urged that the subject application is patentable over Gong, Bak and ICC and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

Respectfully submitted,

DATE:

July 23, 2003

Stephen J. Walder, Jr.

Reg. No. 41,534

Carstens, Yee & Cahoon, LLP

P.O. Box 802334

Dallas, TX 75380

(972) 367-2001

Attorney for Applicants